SÜTRON
visualization | control | communication

# User Manual

## Connection to INTERBUS MMICOM, Interface 06, 14, 18, 21, 76


| | |
|---|---|
| Part Number: | 80 860.659 |
| Version: | 3 |
| Date: | 08.12.2009 |
| Valid for: | TSwin .net 4.x |


visualization | control | communication

| Version | Date | Modifications |
|---|---|---|
| 1 | 26.07.2005 | First edition |
| 2 | 24.11.2005 | Validation extended, chapter "Important Notes" added, overview of function blocks corrected |
| 3 | 08.12.2009 | New Layout, interface 76 added, MPM description FC66 / FC86 |

This manual, including all illustrations contained herein, is copyright protected. Use of this manual by any third party in departure from the copyright provision is forbidden. No part of this manual may be reproduced, translated or electronically or photographically archived or altered without the express written consent from Sütron electronic GmbH. Violations shall be cause for damage liability.
Sütron electronic reserves the right to make any changes that contribute to technical improvement.

# Overall Table of Contents

1 Important Notes ..... 1-1
1.1 Symbols ..... 1-1
1.2 Safety Notes ..... 1-1
1.3 Intended Use ..... 1-1
1.4 Target Group ..... 1-1

2 INTERBUS MMICOM, Interface 06, 14, 18, 21, 76 ..... 2-1
2.1 Integration of the Operating Devices ..... 2-1
2.2 MMICOM Profile ..... 2-1
2.2.1 Direct Process Data Channel ..... 2-1
2.2.2 Indirect Process Data Channel ..... 2-2
2.2.3 Parameter Channel ..... 2-2
2.3 Connecting the Operating Device ..... 2-2
2.3.1 Specification for INTERBUS ..... 2-3
2.4 Data Types ..... 2-4
2.5 Considerations for Optimized Data Transmission ..... 2-4
2.6 Programming ..... 2-7
2.6.1 Protocol Parameters ..... 2-7
2.6.1.1 MMICOM Handshake Timeout ..... 2-7
2.6.1.2 Delay until Connection Set-Up ..... 2-8
2.6.1.3 Floating Point Numbers in the Siemens Format ..... 2-8
2.6.1.4 Optimized Data Transmission ..... 2-8
2.6.2 Polling Area ..... 2-9
2.6.3 Status Messages ..... 2-9
2.6.4 Input Syntax ..... 2-9
2.6.5 Physical Connection ..... 2-11
2.6.5.1 Pin Assignment ..... 2-11
2.6.5.2 2-Wire Remote Bus Cable ..... 2-12
2.6.5.3 Converting from 8-Wire Protocol to 2-Wire Protocol ..... 2-13
2.7 Error Messages ..... 2-14
2.8 Applications ..... 2-16
2.8.1 Siemens S7 Controller ..... 2-16
2.8.1.1 Operating Mode 'Asynchronous' ..... 2-16
2.8.1.2 Operating Mode 'Asynchronous with Synchronization Impulse' ..... 2-16
2.8.1.3 Configuring the Operating Device as an INTERBUS Participant ..... 2-17
2.8.1.4 Phoenix Contact Function Blocks ..... 2-17
2.8.1.5 FC20 (INIT_IB) ..... 2-18
2.8.1.6 FC21 (MEM_READ) ..... 2-22
2.8.1.7 FC22 (MEM_WRIT) ..... 2-23
2.8.1.8 FC28 (IB_SYNC) ..... 2-24
2.8.1.9 Function Blocks from Sütron electronic ..... 2-24
2.8.1.10 Decoding the Variable Number ..... 2-24
2.8.1.11 Overview of Function Blocks from Sütron electronic ..... 2-26
2.8.1.12 Inserting Sources and Function Blocks ..... 2-26

2.8.1.13 FC60 for the Operating Mode 'Asynchronous' with S7-400 ..... 2-28
2.8.1.14 FC66 for the Operating Mode 'Asynchronous' with S7-300 ..... 2-29
2.8.1.15 FC80 for the Operating Mode 'Asynchronous' with S7-400 ..... 2-31
2.8.1.16 FC86 for the Operating Mode 'Asynchronous' with S7-300 ..... 2-32
2.8.1.17 FC65 for the Operating Mode 'Asynchronous with Synchronization Impulse' with S7-300 2-34
2.8.1.18 FC85 for the Operating Mode 'Asynchronous with Synchronization Impulse' with S7-300 2-35
2.8.1.19 Example for the Operating Mode 'Asynchronous' ..... 2-35
2.8.1.20 Example for the Operating Mode 'Asynchronous with Synchronization Impulse' ..... 2-36

A Index ..... A-1

# 1 Important Notes

## 1.1 Symbols

The symbols in this manual are used to draw your attention on notes and dangers.

**Danger**
This symbol is used to refer to instructions which, if ignored or not carefully followed could result in personal injury.

**Note**
This symbol indicates application tips or supplementary notes.

**Reference to source of information**
This symbol refers to detailed sources of information on the current topic.

## 1.2 Safety Notes

- Read this manual carefully before using the software. Keep this manual in a place where it is always accessible to all users.
- The user manual, in particular the safety notes, must be observed by all personnel working with the software and the programmed device.
- Observe the accident prevention rules and regulations that apply to the operating site.
- Installation and operation must only be carried out by qualified and trained personnel.

## 1.3 Intended Use

- The software has to be used for programming operating devices exclusively. Every other use is not permitted.

## 1.4 Target Group

All configuration and programming work in connection with the automation system must be performed by trained personnel only (e.g. qualified electricians, electrical engineers).

The configuration and programming personnel must be familiar with the safety concepts of automation technology.

# 2 INTERBUS MMICOM, Interface 06, 14, 18, 21, 76

To select the correct interface variant in the programming software, take note of the ID on the label attached to the operating device.

The two digits after the slash indicate the interface variant (printed in bold in the example displayed below).

**For example:**

TP32ET/**15**9032

## 2.1 Integration of the Operating Devices

INTERBUS is an open field bus concept that is supported by a number of manufacturers of sensor technology and actuator devices. This means that, in addition to using the same bus to connect to the operating devices, several other participants can be connected to the controller (for example, decentralized inputs and outputs).

The operating device is integrated as a slave in the 2-wire remote bus. In this context, the operating device occupies a data width of 4 words or 8 bytes on the INTERBUS. The controller maps each operating device with 8 bytes IN data and 8 bytes OUT data in the memory map.

If the Physical Addressing operating mode is selected in the controller board, the participants' input and output areas are placed sequentially in the memory of the controller in the same order that they are physically installed in the bus.

If the Logical Address operating mode is selected in the controller board, the location of the input and output areas can be placed freely in the controller memory. The location of these areas is also independent of the sequence of bus participants.

## 2.2 MMICOM Profile

The MMICOM profile distinguishes between the following three basic data channels on the INTERBUS:

1. Direct process data channel
2. Indirect process data channel
3. Parameter channel.

### 2.2.1 Direct Process Data Channel

Direct process data is cyclical data that remains constant while the device is in operation. They are not acknowledged and are incorporated cyclically into the process data channel (for example, 16 inputs of an input module or bit information of a key field - device class A1 or B1).

### 2.2.2 Indirect Process Data Channel

Indirect process data is data that does not remain constant while the device is in operation, and changes depending on external events or requirements. Process data identifiers (PD index) are used to determine the structure of this data. The handshake between both communication partners is defined using a status/control byte.

### 2.2.3 Parameter Channel

In addition to the process data channel, the parameter channel allows an FMS-like background communication to be carried out. In this context, the bytes of a saved log are sequentially transferred via the INTERBUS. This process is usually advantageous for slower communication processes where larger data volumes are being used, and it does not delay the equidistant transfer of time-critical data.

## 2.3 Connecting the Operating Device

In the INTERBUS, the operating device is run in the indirect process data channel. Neither the parameter channel nor the direct process data channel is used. The user data is interpreted in accordance with the MMICOM profile.

The connection to the bus is implemented using dc-decoupled RS485 drivers. The actual bus protocol is handled by the INTERBUS protocol chip.

On the controller side, a manufacturer-specific INTERBUS master controller board is required, preferably with dc-decoupling.

If the controller board in the controller has an 8-wire remote bus interface (25 pin), use the bus terminal IBS 24 BK/LC2 to convert to the 2-wire remote bus (9 pin).

Both the INTERBUS connection and the MMICOM protocol are independent of the controller. As a result, only variable numbers and no real controller addresses are transferred in the MMICOM protocol. The variable number is referenced to the variable in the controller.

A function block must be called in the controller's main program. This function block is assigned the parameters for the location of the IN and OUT data, and must be called for each connected operating device. If necessary, the function block interprets the requests received from the operating device. In this context, either a read or a write function is carried out, which makes the assignment between the variable number and the actual controller variable. If necessary, the data is copied in the controller in an event-controlled manner. As a result, the controller has a say in what data is written and read by the operating device.

All services (requests) required to run the operating device originate in the operating device. The operating device has client functionality. The controller only reacts to the requests of the operating device, and therefore fulfils the functions of a server.

### 2.3.1 Specification for INTERBUS

The following parameters are used to run operating devices on the INTERBUS:

Table 2-1 Parameters for the operating devices on the INTERBUS

| Parameter | | Value |
|---|---|---|
| Width of the Data Channel | Indirect Process Data with Status Word | 4 Words / 8 Bytes |
| | Direct Process Data | None |
| | Parameter Channel | None |
| Identcode | PD Channel with Input and Output Data | $2F_h$ / $47_d$ |
| Function group specification within the MMICOM | B3 | Variable Input |
| | G1 | Variable Request |
| MMICOM Services Used | PD-Index 0x14 | Write Variable 1 Byte (Mandatory) |
| | PD-Index 0x15 | Write Variable 2 Bytes (Mandatory) |
| | PD-Index 0x16 | Write Variable 4 Bytes |
| | PD-Index 0x40 | Read Variable 1 Byte (Mandatory) |
| | PD-Index 0x41 | Read Variable 2 Bytes (Mandatory) |
| | PD-Index 0x42 | Read Variable 4 Bytes |
| | PD-Index 0x84 | Write Bit to Byte |
| | PD-Index 0x85 | Write Bit to Word |

## 2.4 Data Types

The data types specify how many bytes are assigned to a single variable.

Table 2-2 Data types, MMICOM

| Data Type | Access to | Length of Single Variable | Comment |
|---|---|---|---|
| BY | Byte (and Bit also) | 1 Byte | Next Byte is Located 1 Address Higher |
| W | Word | 2 Bytes | Next Word is Located 1 Address Higher |
| W-2 | Word | 2 Bytes | Next Word is Located 2 Address Higher |
| DW | Double Word | 4 Bytes | Next Double Word is Located 1 Address Higher |
| DW-2 | Double Word | 4 Bytes | Next Double Word is Located 2 Address Higher |
| DW-4 | Double Word | 4 Bytes | Next Double Word is Located 4 Address Higher |

The access parameter determines the data length that is evaluated by the operating device, and the MMICOM service used to handle the variable.

**For example:**

A text with ten characters that starts at the variable 100 (access W - word) is made up of the variables 100 to 104.

The following convention applies to data with a data length of more than 4 bytes (for example, alphanumeric texts, tables, polling area, status messages):
Depending on the access type, the data is processed with the corresponding one-, two-, or four-byte access. If there is other data to be processed, it is assumed that this data is located under the next higher variable number in each case.

## 2.5 Considerations for Optimized Data Transmission

The usage of optimized data transmission is not permitted to projects with PC WORX.

To optimize the data transmission and as a result accelerate the speed of screen composition at the operating device you can activate the **Optimized data transmission** in the protocol parameters. In this case the available user data of 4 byte (32 bit) is used more efficient and the number of communication telegrams is reduced. To make this available you have to address the variables consecutively as bit, byte or word access in the corresponding address area.

The following table shows the possible combinations.

Table 2-3 Combinations for optimized data transmission

| DB X Offset Y | DB X Offset Y+1 | DB X Offset Y+2 | DB X Offset Y+3 | Number Of Telegrams | |
|---|---|---|---|---|---|
| | | | | Without Optim. | With Optim. |
| BY | | | | 1 | 1 |
| BY | BY | | | 2 | 1 |
| BY | | BY | | 2 | 2 |
| BY | | | BY | 2 | 1 |
| BY | BY | BY | | 3 | 2 |
| BY | BY | | BY | 3 | 1 |
| BY | | BY | BY | 3 | 1 |
| BY | BY | BY | BY | 4 | 1 |
| W low | W high | | | 1 | 1 |
| W low | W high | W low | W high | 2 | 1 |

The syntax for INTERBUS MMICOM is not controller-specific.
Examples for Siemens S7 controllers:
BY 100, 10 = DB100.DBB10,
W 100, 10 = DB100.DBW10,
DW 100, 10 = DB100.DBD10.
Examples for PC WORX:
BY100 = byte array, byte 100,
W100 = word array, word 100,
DW100 = double word array, double word 100

**BIT access:**

**Without optimized data transmission** up to 8/16/32 bits, which have the same byte-, word- or double word address, are transmitted together.

**With optimized data transmission** also bits which are located in consecutively addressed bytes or words are transmitted together.

Table 2-4 BIT access

| Bit Variables on one Screen | | | | Number of Bits | Number Of Telegrams | |
|---|---|---|---|---|---|---|
| Variable | Address | Low-Bit No | High-Bit No | | Without Optim. | With Optim. |
| 1<br>:<br>8 | BY 100, 10<br>:<br>BY 100, 10 | 0<br>:<br>7 | 0<br>:<br>7 | 8 | 1 | 1 |
| 1<br>:<br>8<br>9<br>:<br>16 | BY 100, 10<br>:<br>BY 100, 10<br>BY 100, 11<br>:<br>BY 100, 11 | 0<br>:<br>7<br>0<br>:<br>7 | 0<br>:<br>7<br>0<br>:<br>7 | 16 | 2 | 1 |
| 1<br>:<br>8<br>9<br>:<br>16<br>17<br>:<br>24<br>25<br>:<br>32 | BY 100, 10<br>:<br>BY 100, 10<br>BY 100, 11<br>:<br>BY 100, 11<br>BY 100, 12<br>:<br>BY 100, 12<br>BY 100, 13<br>:<br>BY 100, 13 | 0<br>:<br>7<br>0<br>:<br>7<br>0<br>:<br>7<br>0<br>:<br>7 | 0<br>:<br>7<br>0<br>:<br>7<br>0<br>:<br>7<br>0<br>:<br>7 | 32 | 4 | 1 |
| 1<br>:<br>16 | W 100, 10<br>:<br>W 100, 10 | 0<br>:<br>15 | 0<br>:<br>15 | 16 | 1 | 1 |
| 1<br>:<br>16<br>17<br>:<br>32 | W 100, 10<br>:<br>W 100, 10<br>W 100, 12<br>:<br>W 100, 12 | 0<br>:<br>15<br>0<br>:<br>15 | 0<br>:<br>15<br>0<br>:<br>15 | 32 | 2 | 1 |
| 1<br>:<br>32 | DW 100, 10<br>:<br>DW 100, 10 | 0<br>:<br>31 | 0<br>:<br>31 | 32 | 1 | 1 |

**BYTE access:**

Table 2-5 BYTE access

| Byte Variables on one Screen | Number of Bytes | Number Of Telegrams | |
|---|---|---|---|
| | | Without Optim. | With Optim. |
| BY 100, 10 | 1 | 1 | 1 |
| BY 100, 10 + BY 100, 11 | 2 | 2 | 1 |
| BY 100, 10 + BY 100, 12 | 2 | 2 | 2 |
| BY 100, 10 + BY 100, 11 + BY 100, 12 | 3 | 3 | 2 |
| BY 100, 10 + BY 100, 11 + BY 100, 13 | 3 | 3 | 1 |
| BY 100, 10 + BY 100, 11 + BY 100, 12 + BY 100, 13 | 4 | 4 | 1 |

For 3 consecutive Bytes 2 accesses are made. The first 2 bytes are accessed with one WORD access and the third byte is accessed with a BYTE access. In case of the third byte is at the end of a data block this will not cause a communication error.

**WORD access:**

Table 2-6 WORD access

| Word Variables on one Screen | Number of Words | Number Of Telegrams | |
|---|---|---|---|
| | | Without Optim. | With Optim. |
| W 100, 10 | 1 | 1 | 1 |
| W 100, 10 + W 100, 12 | 2 | 2 | 1 |

## 2.6 Programming

### 2.6.1 Protocol Parameters

With the protocol parameters, you can adapt the communication of the controller used.

#### 2.6.1.1 MMICOM Handshake Timeout

This parameter specifies how long the operating device waits for an acknowledgement from the controller.

Table 2-7 MMICOM handshake timeout

| Configurable Values | Default Value |
|---|---|
| 0 ms to 65535 ms | 100 ms |

#### 2.6.1.2 Delay until Connection Set-Up

This parameter specifies the waiting time after which the operating device starts the communication.

Table 2-8 Delay until connection set-up

| Configurable Values | Default Value |
|---|---|
| 0 s to 20 s | 10 s |

#### 2.6.1.3 Floating Point Numbers in the Siemens Format

This parameter specifies whether floating point numbers are exchanged in the Siemens-specific format or IEEE format.

Table 2-9 Floating point number in the Siemens format

| Configurable Values | Default Value |
|---|---|
| IEEE Format | |
| Siemens Format | X |

#### 2.6.1.4 Optimized Data Transmission

This parameter specifies whether optimized data transmission should be used. For using optimized data transmission it is recommended to fulfill certain considerations while programming the controller!

Table 2-10 Optimized data transmission

| Configurable Values | Default Value |
|---|---|
| ON | |
| OFF | X |

See chapter “Considerations for Optimized Data Transmission“ on page 2-4.

### 2.6.2 Polling Area

The polling area is used to manage the write coordination byte (WCB), the serial message channel and the LEDs in the function keys. This area is continuously polled by the operating device. The polling area is structured as follows:

Table 2-11 Additional function - polling area, MMICOM

| Bit Number | 15 | 8 | 7 | 0 |
|---|---|---|---|---|
| Variable X | WCB | | Reserved | |
| Variable X+1 | Serial Message Channel High Byte | | Serial Message Channel Low Byte | |
| Variable X+2 | LED 1 to 4 | | LED 5 to 8 | |
| Variable X+3 | LED 9 to 12 | | LED 13 to 16 | |
| Variable X+4 | LED 17 to 20 | | LED 21 to 24 | |
| Variable X+5 | LED 25 to 28 | | LED 29 to 32 | |

### 2.6.3 Status Messages

Status messages are the static assignment of flags (bits) in the controller to plain text messages in the operating device.

The area for the status messages has the following structure:

Table 2-12 Status messages, MMICOM

| Bit Number | 15 | 8 | 7 | 0 |
|---|---|---|---|---|
| Variable X | Message 16 to 9 | | Message 8 to 1 | |
| Variable X+1 | Message 32 to 25 | | Message 24 to 17 | |
| Variable X+2 | Message 48 to 41 | | Message 40 to 33 | |

### 2.6.4 Input Syntax

The following image illustrates the structure of the input syntax for variables in the programming software.

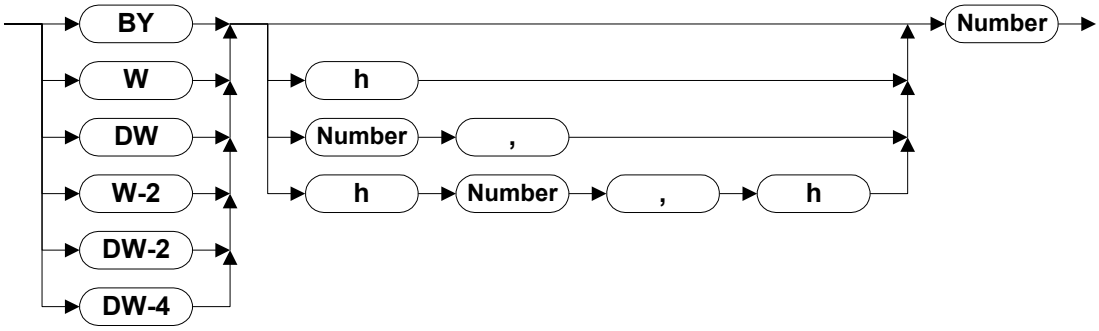



Figure 2-1 Syntax diagram INTERBUS MMICOM

In the programming software the **Variable** dialog opens, to define a controller variable.

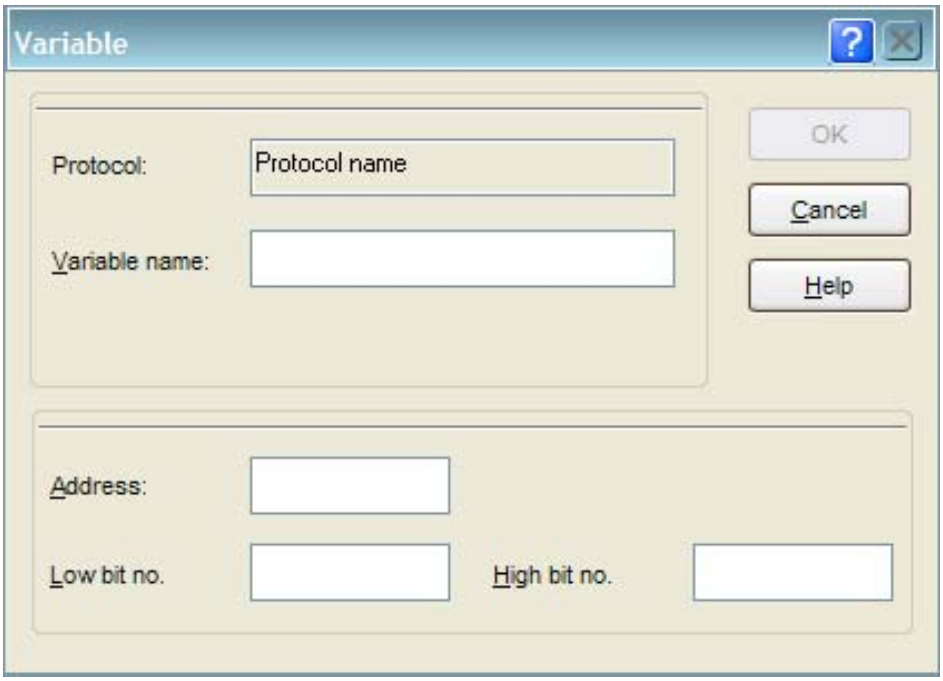



Figure 2-2 Variable dialog

**Variable name:**

Enter any name for a variable in this field.

**Address:**

Enter an address according to the syntax diagram above.

**Low-Bit No. and High-Bit No.:**

If you specify **a single bit** of a byte, word or double word, enter the same bit number in each field.

**Example:**

To specify the address of the data block 100, byte 10, bit 0 enter the following values.
Address = BY100,10
Low-Bit No. = 0
High-Bit No. = 0

If you specify the address of **several bits** (bitstream) of one byte, word or double word, enter the lower significant bit number in the Low-Bit No. field and the higher significant bit number in the High-Bit No. field.

**Example:**

To specify the address of the data block 100, byte 10, bits 0 to 3 enter the following values.
Address = BY 100,10
Low-Bit No. = 0
High-Bit No. = 3

The syntax for INTERBUS MMICOM is not controller-specific.
Examples for Siemens S7 controllers:
BY 100, 10 = DB100.DBB10,
W 100, 10 = DB100.DBW10,
DW 100, 10 = DB100.DBD10.
Examples for PC WORX:
BY100 = byte array, byte 100,
W100 = word array, word 100,
DW100 = double word array, double word 100

## 2.6.5 Physical Connection

### 2.6.5.1 Pin Assignment

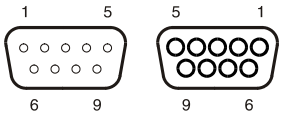



Figure 2-3 9 pin D-SUB male connector strip and female connector strip

Connector in the terminal: 9 pin D-SUB male connector strip for remote bus in.

Table 2-13 Pin assignment remote bus in (INTERBUS)

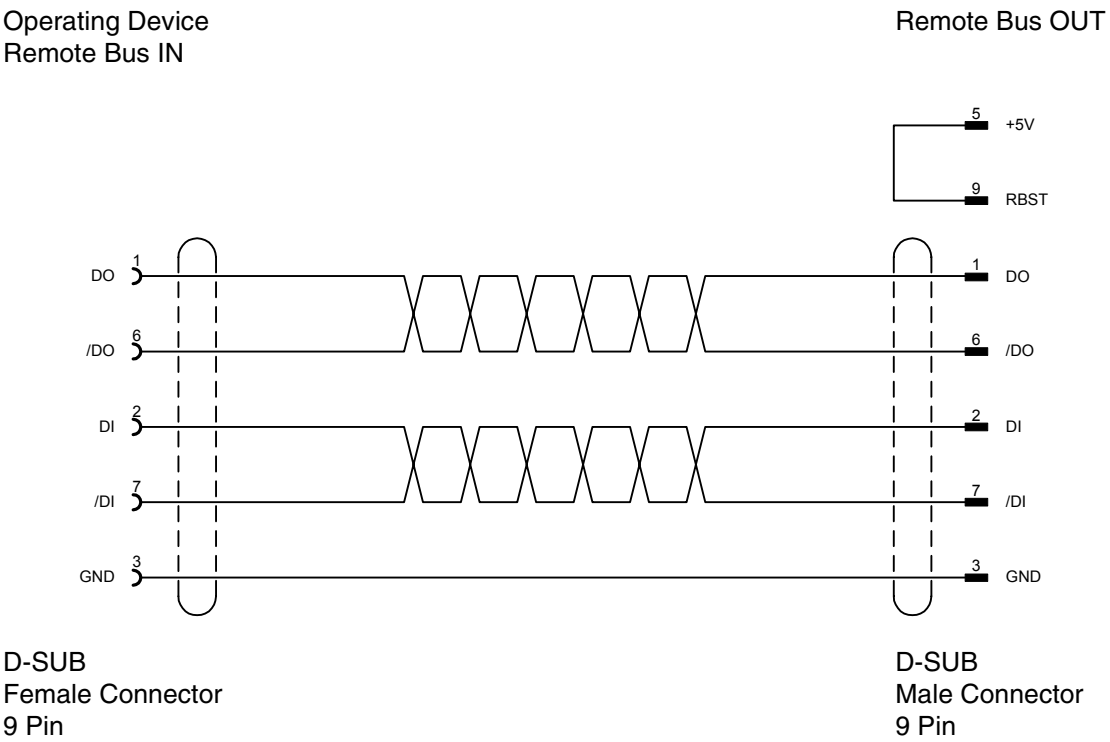
| Pin | Designation | Function |
|---|---|---|
| 1 | DO | Data Input |
| 2 | DI | Data Output |
| 3 | GND | Ground |
| 4 | nc | Not Connected |
| 5 | nc | Not Connected |
| 6 | /DO | Data Input, Inverted |
| 7 | /DI | Data Output, Inverted |
| 8 | nc | Not Connected |
| 9 | nc | Not Connected |

Connector in the terminal: 9 pin D-SUB female connector strip for remote bus out.

Table 2-14 Pin assignment remote bus out (INTERBUS)

| Pin | Designation | Function |
|---|---|---|
| 1 | DO | Data Output |
| 2 | DI | Data Input |
| 3 | GND | Ground |
| 4 | nc | Not Connected |
| 5 | +5 V | Power Supply +5 VDC |
| 6 | /DO | Data Output, Inverted |
| 7 | /DI | Data Input, Inverted |
| 8 | nc | Not Connected |
| 9 | RBST | Remote Bus Status |

#### 2.6.5.2 2-Wire Remote Bus Cable

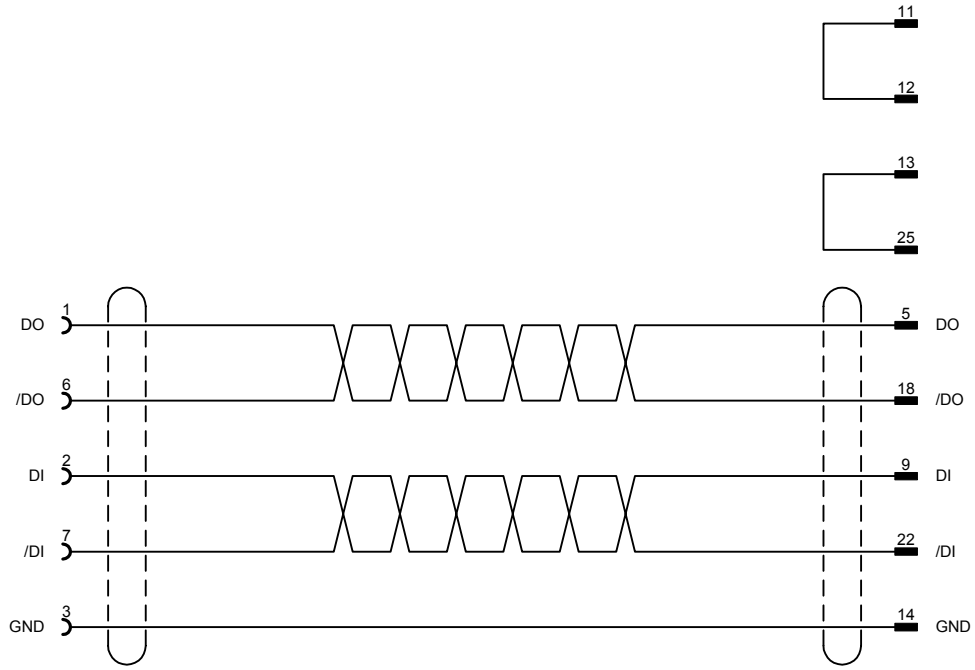


#### 2.6.5.3 Converting from 8-Wire Protocol to 2-Wire Protocol

Using bus terminal IBS 24 BK.

Operating Device
Remote Bus IN

Remote Bus OUT




D-SUB
Female Connector
9 Pin

D-SUB
Male Connector
25 Pin

## 2.7 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

Code XXXXX

Subcode XXXXX

Retries XXXXX

Table 2-15 Error messages INTERBUS MMICOM, interface 06, 14, 18, 21, 76

| Code | Subcode | Error Type | Possible Cause |
|---|---|---|---|
| 2 | | The controller has transferred data to the operating device, but the operating device has not requested the data. The subcode specifies the received PD index in decimal format. | |
| 40 | | Illegal system variable. | The project contains an illegal system variable. |
| 50 | | Malfunction bit not set by controller.The subcode specifies the received PD Index in decimal format. | |
| 51 | | Online bit not set by controller.The subcode specifies the received status control byte in decimal format. | Bus is not running, PLC program is missing or controller is in STOP mode. |
| 52 | | Standard bit and/or index bit not set by controller.The subcode specifies the received status control byte in decimal format. | Bus is not running, PLC program is missing or controller is in STOP mode. |
| 53 | | Wrong PD index received. The subcode specifies the received PD Index in decimal format. | |
| 54 | | Handshake error (receive timeout). The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | PLC program is missing or controller is in STOP mode or the function block for the operating device is not active. |
| 55 | | Handshake error (send timeout). The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | PLC program is missing or controller is in STOP mode. |
| 56 | | Variable has wrong base size. The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | Wrong access type specified. |
| 57 | | Handshake error. The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | Handshake bits were not set correctly by controller. |

Table 2-15 Error messages INTERBUS MMICOM, interface 06, 14, 18, 21, 76

| Code | Subcode | Error Type | Possible Cause |
|---|---|---|---|
| 58 | | Access error. The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | No valid data cycles are executed on the INTERBUS. |
| 59 | | Wrong variable number received. The subcode indicates the variable number - in decimal format - which was being edited when the error occurred. | The 8 byte user data are not consistent. There is an error within the controller board. |
| 60 | | Wrong PD index received. The subcode specifies the received PD Index in decimal format. | The 8 byte user data are not consistent. There is an error within the controller board. |

By setting up the following system variables in one of the screens in the project, you can log the error messages with the operating device.

– **ComParityCount** to log the number of the error codes 59.
– **ComOverrunCount** to log the number of the error codes 60.
– **ComFrameCount** to log the number of all other error codes.

# 2.8 Applications

## 2.8.1 Siemens S7 Controller

### 2.8.1.1 Operating Mode 'Asynchronous'

For the controllers S7-300 and S7-400, the controller boards are normally run in the asynchronous mode of operation.

To set the operating mode with IBS CMD G4, carry out the following steps:

1. Select the controller board.
2. From the context menu, select **Settings**.

The **Controller Board Settings** dialog opens.

3. Open the **Bus Operation** tab.
4. From the **Bus Operation** field, select the **Asynchronous** operating mode.
5. Click **OK** to confirm your settings.

### 2.8.1.2 Operating Mode 'Asynchronous with Synchronization Impulse'

You can operate the INTERBUS on both the S7-300 and the S7-400 in the **asynchronous with synchronization impulse** operating mode.

It is, however, recommended to use the S7-400 in the **asynchronous** mode of operation only.

To set the operating mode 'Asynchronous with Synchronization Impulse", carry out the following steps:

1. Select the controller board.
2. From the context menu, select **Settings**.

The **Controller Board Settings** dialog opens.

3. Open the **Bus Operation** tab.
4. From the **Bus Operation** field, select the operating mode **Asynchronous with Synchronization Impulse**.
5. Click the **Bus Operation Options** button.

The **Bus Operation Options** dialog opens.

6. Select the **Board's Interrupt Mode** check box.

This ensures that the parameters for the process alarm, interrupt line 1 and alarm OB 40 are transferred to the controller system.

7. Click **OK** to confirm your settings.

To parameterize data sets to transfer the process data, carry out the following steps:

1. Select the controller board.
2. From the context menu, select **Settings**.

The **Controller Board Settings** dialog opens.

3. Open the **IBS Data Records** tab.

The **IBS Data Records** tab allows you to specify the following parameters:

Table 2-16 Parameters for IBS data sets

| Parameter | Description |
|---|---|
| Record | Choose the number of the data set you want to specify the parameters for. |
| Area | Specify the destination area (DB, E, M) or the source area (A, DB, M), respectively, in the controller. |
| Data Block | If you selected DB for the destination/source area, enter the number of your target/source data block. |
| Offset (Byte) | Enter the start address of the data to be transferred into the selected target/source area. |
| Length (Byte) | Enter the length of the data to be transferred (8 bytes for operating devices). |

4. Click **OK** to confirm your settings.

#### 2.8.1.3 Configuring the Operating Device as an INTERBUS Participant

To add an operating device to the INTERBUS, carry out the following steps:

1. Select the controller board.
2. From the context menu, select **Insert with ID Code**.

The **Insert Device** dialog opens.

3. Enter the **ID code** in either the hexadecimal (**2F**) or decimal format (**47**).
4. Enter the value **8** for the **Process Data Channel**.
5. For the **Device Type** select the **Remote Bus Device** button.
6. Click OK to confirm your input.

The **Device Description** dialog opens. Here you can enter a description of the operating device.

7. Click **OK** to confirm this dialog.

The configuration is complete. Save the configuration to the parameterization memory.

#### 2.8.1.4 Phoenix Contact Function Blocks

Phoenix Contact provides the following functions and data blocks for the operating mode 'Asynchronous with Synchronization Impulse':

– FC20 (INIT_IB),
– FC21 (MEM_READ),
– FC22 (MEM_WRIT),
– FC24 (IB_DIAG),
– FC28 (IB_SYNC) and
– DB20 (IBDB).

When operating the S7-300 controller in the operating mode **Asynchronous**, you need the INTERBUS data block also.

#### 2.8.1.5 FC20 (INIT_IB)

The FC20 function is called-up once when the controller (OB100) is started-up. It controls the start-up behavior of the controller board and initializes the INTERBUS data block (IBDB, DB20 in the example below) with the required data. The settings for timer T1 are fixed.

**Parameter:**

The busy bit must be set when the function is called up. Otherwise no action will be taken.

```
UN #FC20BusyBit
S  #FC21BusyBit
CALL FC 20
IBDB            :=  DB20
COM_ADR         :=  256
DIAG_STATE      :=  0
DIAG_PARA       :=  0
FKN_START       :=  0
FKN_PARA        :=  0
FKN_STATE       :=  0
MEM_READ        :=  21
MEM_WRITE       :=  22
LOAD            :=  0
BOOT            :=  0
MODE            :=  0
TIMER_NR        :=  T1
SOURCE          :=  0
CONFIGURATION   :=  DW#16#8000000F
RET             :=  #FC20RetBit
BUSY            :=  #FC20BusyBit
```

Table 2-17 Parameters for the FC20 function

| Parameter | Value | Description |
|---|---|---|
| IBDB | DB20 | Number of the global INTERBUS data block.<br>This is the data block in which the FC20 function specifies the settings. |
| COM_ADR | | S7 base address of the controller board |
| DIAG_STATE | 0 | S7 address for the diagnostic status register.<br>This address depends on the controller board's base address and is assigned automatically. Therefore, enter the value 0 (zero). |
| DIAG_PARA | 0 | S7 address for the diagnostic parameter register.<br>This address depends on the controller board's base address and is assigned automatically. Therefore, enter the value 0 (zero). |
| FKN_START | 0 | S7 address for the standard function start register.<br>This address depends on the controller board's base address and is assigned automatically. Therefore, enter the value 0 (zero). |
| FKN_PARA | 0 | S7 address for the standard function parameter register.<br>This address depends on the controller board's base address and is assigned automatically. Therefore, enter the value 0 (zero). |

Table 2-17 Parameters for the FC20 function

| Parameter | Value | Description |
|---|---|---|
| FKN_STATE | 0 | S7 address for the standard function status register. This address depends on the controller board's base address and is assigned automatically. Therefore, enter the value 0 (zero). |
| MEM_READ | | Block number (21) for the read driver FC21 (MEM_READ) |
| MEM_WRITE | | Block number (22) for the write driver FC22 (MEM_WRITE) |
| LOAD | | The interface's module configuration is loaded. |
| | 0 (Flash) | The configuration is loaded from the parameterization memory. This requires programming through the IBS CMD/parameterization memory. |
| | 1 (APPLIC) | The configuration is loaded through the application's function blocks. |
| | 2 (CMD_IB) | The configuration is loaded via a CMD file and data blocks. |
| IN_BOOT | | This parameter controls the INTERBUS start-up (only applies if LOAD = 0). |
| | 0 | The bus system is started from the configuration (the parameterization stored on the Flash card contains the bus system start-up information). |
| | 1 | The bus system is started with the configuration frame 1 via the initialization driver. |
| MODE | | Operating mode (BA) of the controller board |
| | 0 (A_SYN) | Asynchronous |
| | 1 (A_SYN_IMP) | Asynchronous with synchronization impulse acting as an interrupt |
| TIMER_NR | | Number of the timer used by the function |
| SOURCE | | Number of the parameterization DB for LOAD = 2 |

Table 2-17 Parameters for the FC20 function

| Parameter | Value | Description |
|---|---|---|
| CONFIGURATION | | 32 configuration bits for INTERBUS blocks |
| | Bit 0 = 0 | Peripheral entries (PF entries) are made by the function IB_DIAG in the IBDB. |
| | Bit 0 = 1 | Peripheral entries (PF entries) are not made by the function IB_DIAG in the IBDB. |
| | Bit 1 = 0 | User error entries (USER entries) are made by the function IB_DIAG in the IBDB. |
| | Bit 1 = 1 | User error entries (USER entries) are not made by the function IB_DIAG in the IBDB. |
| | Bit 2 = 0 | Bus error entries (BUS entries) are made by the function IB_DIAG in the IBDB. |
| | Bit 2 = 1 | Bus error entries (BUS entries) are not made by the function IB_DIAG in the IBDB. |
| | Bit 3 = 0 | Controller entries (CTRL entries) are made by the function IB_DIAG in the IBDB. |
| | Bit 3 = 1 | Controller entries (CTRL entries) are not made by the function IB_DIAG in the IBDB. |
| | Bit 4 = 0 | Reserved |
| | Bit 4 = 1 | Reserved |
| | Bit 31 = 0 | Not relevant |
| | Bit 31 = 1 | This bit must always be set. This informs all of the driver blocks that this is a S7-300 DSC controller board. |
| | DW#16#8000000F | No errors are entered into the IBDB. |
| | DW#16#80000000 | All errors are entered into the IBDB. |

Table 2-17 Parameters for the FC20 function

| Parameter | Value | Description |
|---|---|---|
| RET | | When the result bit is set (1), DBW56 of the IBDB contains an error code with the following meaning:<br>High-Byte DBB56 = error number<br>Low-Byte DBB57 = number of the defective step |
| | Error number 1 | Timeout |
| | Error number 2 | Negative result for 'Stop and reconfiguration request' |
| | Error number 3 | Negative result for 'Start data transfer' |
| | Error number 4 | Parameter 'LOAD' invalid. |
| | Error number 5 | Communication register cannot be read (basic address is set incorrectly or hardware defective) |
| | Error number 6 | Error in parameterization sequence (SOURCE parameter):<br>– The diagnostic status register displayed is entered in the IBDB at DBW 76.<br>– The diagnostic parameter register displayed is entered in the IBDB at DBW 78.<br>– The incorrect service code is entered in the IBDB at DBW 74 and is used for internal purposes. |
| BUSY | 1 | Function active. This bit must be set during call-up. |

#### 2.8.1.6 FC21 (MEM_READ)

The function FC21 (MEM_READ) reads data from the controller board and copies it to the specified destination area of the controller.

**Parameter:**

The number of the INTERBUS data block (IBDB, in the following example: DB20) is transferred as a parameter. The busy bit must be set when the function is called up. Otherwise no action will be taken.

```
UN #FC21BusyBit
S  #FC21BusyBit
CALL FC21
IBDB                :=  DB20
MODE                :=  1
SOURCE              :=  0
DEST_AREA           :=  0
DEST_AREA_NR        :=  0
DEST_OFFSET         :=  0
DEST_LENGTH         :=  0
RET                 :=  #FC21RetBit
BUSY                :=  #FC21BusyBit
```

Table 2-18 Parameters for the FC21 function

| Parameter | Value | Description |
|---|---|---|
| IBDB | DB20 | Number of the global INTERBUS data block. Here it is DB20. |
| MODE | 1 | The function's operating mode.Mode 1 here (Read from data set 1). This must be parameterized in the IBS CMD. |
| SOURCE | 0 | Data source. Not relevant in mode 1. Always 0. |
| DEST_AREA | 0 | Data destination area. Not relevant in mode 1. Always 0. |
| DEST_AREA_NR | 0 | Number of the data block if the destination area = data.Not relevant in mode 1. Always 0. |
| DEST_OFFSET | 0 | Start address in the destination area (DEST_AREA). Not relevant in mode 1. Always 0. |
| DEST_LENGTH | 0 | Number of words to be read. Not relevant in mode 1. Always 0. |
| RET | | Result bit. When the result bit is set, the DBW58 of the IBDB contains an error code with the following meaning: DBW58 = error number 1 to 5. |
| | Error number = 1 | The DEST_AREA_NR parameter has been used to specify 'DB0' as the destination. |
| | Error number = 2 | The DEST_AREA parameter has been used to specify an invalid destination area. |
| | Error number = 3 | The MODE parameter has been used to specify an invalid operating mode. |
| | Error number = 4 | An invalid CR number (number for a PCP-enabled INTERBUS participant) has been specified in the parameter SOURCE. |
| | Error number = 5 | A data set not enabled (activated) in IBS CMD has been specified. |
| Busy | 1 | Function active. This bit must be set during call-up. |

#### 2.8.1.7 FC22 (MEM_WRIT)

The function FC22 writes data from the controller's source area to the specified destination area on the controller board.

**Parameter:**

The number of the INTERBUS data block (IBDB, in the following example: DB20) is transferred as a parameter. The busy bit must be set when the function is called up. Otherwise no action will be taken.

```
UN #FC22BusyBit
S  #FC22BusyBit
CALL FC 22
IBDB            :=  DB20
MODE            :=  1
SOURCE_AREA     :=  0
SOURCE_AREA_NR  :=  0
SOURCE_OFFSET   :=  0
SOURCE_LENGTH   :=  0
DESTINATION     :=  0
RET             :=  #FC22RetBit
BUSY            :=  #FC22BusyBit
```

Table 2-19 Parameters for the FC22 function

| Parameter | Value | Description |
|---|---|---|
| IBDB | DB20 | Number of the global INTERBUS data block. Here it is DB20. |
| MODE | 1 | The function's operating mode. Here: mode 1 (writing to data set 1).This must be parameterized in IBS CMD. |
| SOURCE_AREA | 0 | Data source. Not relevant in mode 1. Always 0. |
| SOURCE_AREA_NR | 0 | Number of the data block if the source area = data. Not relevant in mode 1. Always 0. |
| SOURCE_OFFSET | 0 | Start address in the SOURCE_AREA. Not relevant in mode 1. Always 0. |
| SOURCE_LENGTH | 0 | Number of words to be written. Not relevant in mode 1. Always 0. |
| DESTINATION | 0 | Data destination. Not relevant in mode 1. Always 0. |
| RET | | Result bit. When the result bit is set, the DBW60 of the IBDB contains an error code with the following meaning: DBW60 = error number 1 to 5. |
| | Error number = 1 | The SOURCE_AREA_NR parameter has been used to specify 'DB0' as the source. |
| | Error number = 2 | The SOURCE_AREA parameter has been used to specify an invalid source area. |
| | Error number = 3 | The MODE parameter has been used to specify an invalid operating mode. |
| | Error number = 4 | An invalid CR number (number for a PCP-enabled INTERBUS participant) has been specified in the parameter DESTINATION. |
| | Error number = 5 | A data set not enabled (activated) in IBS CMD has been specified. |
| Busy | 1 | Function active. This bit must be set during call-up. |

#### 2.8.1.8 FC28 (IB_SYNC)

The function FC28 controls the protocol in the operating mode 'Asynchronous with Synchronization Impulse' and monitors the transfer of consistent data greater than 16 bit. In addition, it requires functions for reading and writing I/O data (FC50 and FC51 in the following example). These functions must be programmed without parameters.

**Parameter:**

```
Call FC 28
IBDB:=        DB20
READ_FC:=     FC50
WRITE_FC:=    FC51
RET:=        M10.1
```

Table 2-20 Parameters for the FC28 function

| Parameter | Value | Description |
|---|---|---|
| IBDB | DB20 | Number of the global INTERBUS data block. Here it is DB20. |
| READ_FC | | Function to be called to read the input data. |
| WRITE_FC | | Function to be called up to write the output data. |
| RET | 0 | Function was executed with no errors. |
| | 1 | Error. The function was exited. |

#### 2.8.1.9 Function Blocks from Sütron electronic

Unlike an S5 controller, the S7 controller has a byte structure in all memory areas. This means that you must use the following designations for variables with more than four bytes in the controller:

W-2 instead of W for:

– The polling area
– The parallel message system
– The time update from and to the controller
– The recipe buffer

DW-4 instead of DW for:

– Alphanumeric variables; The length of the variable must be divisible by four.

If a variable with a memory space requirement of more than four bytes is inadvertently assigned the incorrect data ID, the user data is incorrectly interpreted. The variable does not issue any correct values.

Caution:

When you are starting up the operating device, a one-byte read access is carried out on the variable with the number zero in a test communication. The data read is not interpreted, that is, the read function block does not have to decode this variable number.

#### 2.8.1.10 Decoding the Variable Number

The function blocks' write and read function decodes the received variable number as follows:

**For example:** W h 4BC8

| 4 | | | | B | | | | C | | | | 8 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data Block Number | | | | | | | | Byte/word/double-word number | | | | | | | |

| Abbreviation | Value | Description |
|---|---|---|
| W | | Data Width |
| | DW | Double Word = 32 Bits |
| | DW-4 | Double Word = 32 Bits |
| | W | Word = 16 Bits |
| | W-2 | Word = 16 Bits |
| | BY | Byte = 8 Bits |
| | W | Bit access to word address (if bit address is specified) |
| | BY | Bit access to byte address (if bit address is specified) |
| h | | Hexadecimal Number |
| 4BC8 | | Numeric Code for Data ID<br>4B = Data Block Number (here: DB75)<br>C8 = Double-Word/Word/Byte Number (here: Word 200) |

### 2.8.1.11 Overview of Function Blocks from Sütron electronic

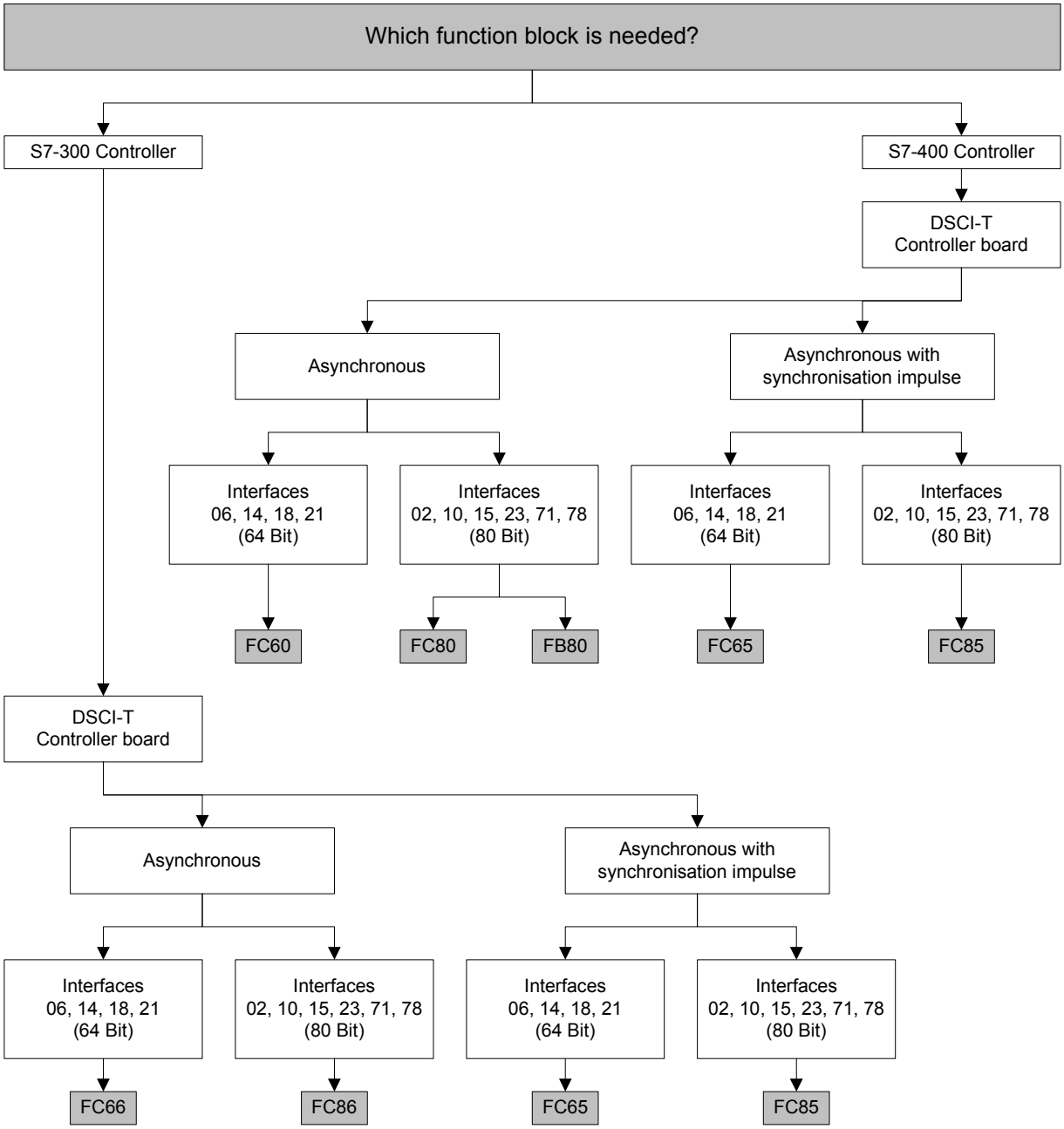



Figure 2-4 Overview of function blocks from Sütron electronic

### 2.8.1.12 Inserting Sources and Function Blocks

The supplied Sütron STL files are imported into the SIMATIC Manager using the entry **Sources**. To do this, select the menu item **External Source** from the menu **Insert**.

The function blocks required depend on the controller type and the connection of the operating device (see diagram).

The files are located in the programming software directory under:

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [02,10,15,23,71,78]\300 ASYNC

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [02,10,15,23,71,78]\300_400 ASYMSI

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [02,10,15,23,71,78]\400 ASYNC

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [06,14,18,21]\300 ASYNC

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [06,14,18,21]\300_400 ASYMSI

…\FBs\INTERBUS\SIEMENS\S7\MMI COM Schnittstelle [06,14,18,21]\400 ASYNC

The numbers in the square brackets indicate the operating device's interface type.

The imported STL sources must subsequently be converted to the S7 function block format using the STL Editor.

Sütron's functions and function blocks use the register-indirect, cross-area addressing technique. If interrupt-controlled program calls are permitted in the program cycle, the contents of the address registers 1 and 2, and possibly also the local data stack, may need to be saved, in addition to the contents of both accumulators. For more information, see the section on how to avoid errors when calling function blocks (Avoiding Errors when Calling Blocks)" in the programming manual for your S7 controller (system software S7-300/400, program design).

The operating device can access the controller data on a bit, byte, word, or double-word basis. Bit access is possible on a byte or word address. As the MMICOM profile itself does not transfer data type IDs, these are transferred using the variable numbers. Each variable is uniquely identified using a variable number. This number is defined during programming in the programming software.

The functions must be called cyclically in the application program. A separate function block must be called for each connected operating device.

#### 2.8.1.13 FC60 for the Operating Mode 'Asynchronous' with S7-400

This function decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 8 bytes. To optimize the transfer time, the function block directly accesses the peripheral addresses and not the process image.

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

**Parameter:**

```
Call FC 60
PIN:=      P#P10.0
POUT:=     P#P10.0
DEST:=P#DB101.DBX0.0 Byte 8
SOURCE:=P#DB101.DBX8.0 Byte 8
VAROFFSET:=      0
DBOFFSET:=       0
```

Table 2-21 Parameters for the FC60 function

| Parameter | Value | Description |
|---|---|---|
| PIN | P#P10.0 | Start address of the P inputs of the operating device (Example: P#P10.0 = PEB10) |
| POUT | P#P10.0 | Start address of the P outputs of the operating device (Example: P#P10.0 = PAB10) |
| DEST | | Memory address of the input data. This address must NOT be overwritten by other program parts. |
| SOURCE | | Memory address of the output data. This address must NOT be overwritten by other program parts. |
| DBOFFSET | | DB offset which is added to the data block address of the variables. |
| VAROFFSET | | Variable offset which is added to the byte address of the variables. |

#### 2.8.1.14 FC66 for the Operating Mode 'Asynchronous' with S7-300

This function decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 8 bytes.

Because this block accesses the MPM (Multi Port Memory) of the controller board, the operating device must be parametrized with the INTERBUS project tool CMD into a INTERBUS data set of its own.
This INTERBUS data set may not be updated with commands of FC21 (MEM_READ) / FC22 (MEM_WRIT) within the OB1 on the I/O range.
The calculation of the parameters PIN / POUT is carried out with the following formula:

```
PIN/POUT = MPM_offset_Interbus_dataset +
( CMD_processdata_assignment_IBS_participant - CMD_Interbus_dataset_offset )

Explanation:
For the INTERBUS data sets the MPM_offset_Interbus_dataset is defined as follows:
DS1 = 0 / DS2 = 128 / DS3 = 256 / DS4 = 384 / DS5 = 512 (all values decimal)
CMD_processdata_assignment_IBS_participant:
See CMD-Project: Device --> process data --> parameter assignments
CMD_Interbus_dataset_offset:
See CMD-Project: Controller board --> settings --> IBS data records for inputs
/ outputs --> parameter offset

Example (all values are decimal):
A operating device in INTERBUS data set 3
MPM_offset_Interbus_dataset = 256
CMD_Interbus_dataset_offset = 256
CMD_processdata_assignment_IBS_participant = 256 for I/O
PIN = POUT = 256 + (256 - 256) = 256
```

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

**Parameter:**

```
Call FC 66
PIN:=           10
POUT:=          10
IBDB:=        DB20
DEST:= DB10.DBX0.0
SOURCE:=DB10.DBX8.0
VAROFFSET:=      0
DBOFFSET:=       0
```

Table 2-22 Parameters for the FC66 function

| Parameter | Value | Description |
|---|---|---|
| PIN | 10 | Offset within the INTERBUS input data (start address of the operating device's input data)<br>(Example: 10 = E10) |
| POUT | P#P10.0 | Start address of the P outputs of the operating device<br>(Example: P#P10.0 = PAB10) |
| IBDB | DB20 | INTERBUS data block |
| DEST | | Destination address of the operating device's data in the controller |

Table 2-22 Parameters for the FC66 function

| Parameter | Value | Description |
|---|---|---|
| SOURCE | | Source address of the operating device's data in the controller |
| DBOFFSET | | DB offset which is added to the data block address of the variables. |
| VAROFFSET | | Variable offset which is added to the byte address of the variables. |

#### 2.8.1.15 FC80 for the Operating Mode 'Asynchronous' with S7-400

An instance data block must be specified when the function block is called up. The protocol data for inputs and outputs stored in this block are only visible to the FB88.

This function block decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 10 bytes. The function block access the peripheral addresses directly.

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

Each of the available I/O points of the module are mapped to byte position 10 in the IN and OUT frame of the module and are not influenced by the function block.

**Parameter:**

```
Call FB 80, DB101
P_IN:=          10
P_OUT:=         10
DBOFFSET:=       0
VAROFFSET:=      0
```

Table 2-23 Parameters for the FC80 function

| Parameter | Value | Description |
|---|---|---|
| P_IN | 10 | Start address of the operating device's inputs (INTEGER)<br>(Example: 10 = PEB10) |
| P_OUT | 10 | Start address of the operating device's outputs<br>(Example: 10 = PAB10) |
| IBDB | DB20 | INTERBUS data block |
| DEST | | Destination address of the operating device's data in the controller |
| SOURCE | | Source address of the operating device's data in the controller |
| DBOFFSET | | DB offset which is added to the data block address of the variables. |
| VAROFFSET | | Variable offset which is added to the byte address of the variables. |

#### 2.8.1.16 FC86 for the Operating Mode 'Asynchronous' with S7-300

This function decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 10 bytes.

Because this block accesses the MPM (Multi Port Memory) of the controller board, the operating device must be parametrized with the INTERBUS project tool CMD into a INTERBUS data set of its own.
This INTERBUS data set may not be updated with commands of FC21 (MEM_READ) / FC22 (MEM_WRIT) within the OB1 on the I/O range.
The calculation of the parameters PIN / POUT is carried out with the following formula:

```
PIN/POUT = MPM_offset_Interbus_dataset +
( CMD_processdata_assignment_IBS_participant - CMD_Interbus_dataset_offset )

Explanation:
For the INTERBUS data sets the MPM_offset_Interbus_dataset is defined as follows:
DS1 = 0 / DS2 = 128 / DS3 = 256 / DS4 = 384 / DS5 = 512 (all values decimal)
CMD_processdata_assignment_IBS_participant:
See CMD-Project: Device --> process data --> parameter assignments
CMD_Interbus_dataset_offset:
See CMD-Project: Controller board --> settings --> IBS data records for inputs
/ outputs --> parameter offset

Example (all values are decimal):
A operating device in INTERBUS data set 3
MPM_offset_Interbus_dataset = 256
CMD_Interbus_dataset_offset = 256
CMD_processdata_assignment_IBS_participant = 256 for I/O
PIN = POUT = 256 + (256 - 256) = 256
```

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

**Parameter:**

```
Call FC 86
PIN:=           10
POUT:=          10
IBDB:=        DB20
DEST:= DB10.DBX0.0
SOURCE:=DB10.DBX10.0
DBOFFSET:=       0
VAROFFSET:=      0
```

Table 2-24 Parameters for the FC86 function

| Parameter | Value | Description |
|---|---|---|
| PIN | 10 | Offset within the INTERBUS input data (start address of the operating device's input data)<br>(Example: 10 = E10) |
| POUT | 10 | Offset within the INTERBUS output data (start address of the operating device's output data)<br>(Example: 10 = PAB10) |
| IBDB | DB20 | INTERBUS data block<br>(Example: DB20) |
| DEST | DB10.DBX0.0 | Destination address of the operating device's data in the controller<br>(Example: DB10.DBX0.0) |

Table 2-24 Parameters for the FC86 function

| Parameter | Value | Description |
|---|---|---|
| SOURCE | DB10.DBX10.0 | Source address of the operating device's data in the controller<br>(Example: DB10.DBX10.0) |
| DBOFFSET | 0 | DB offset which is added to the data block address of the variables<br>(Example: 0) |
| VAROFFSET | 0 | Variable offset which is added to the byte address of the variables<br>(Example: 0) |

#### 2.8.1.17 FC65 for the Operating Mode 'Asynchronous with Synchronization Impulse' with S7-300

This function decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 8 bytes. Accesses are made to the data sets which were parameterized in IBS CMD.

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

**Parameter:**

```
Call FC 65PIN   :=  EB10POUT:=AB10DBOFFSET:=0VAROFFSET:=0
```

Table 2-25 Parameters for the FC65 function

| Parameter | Value | Description |
|---|---|---|
| PIN | EB10 | Start address of the operating device's inputs<br>(Example: EB10) |
| POUT | AB10 | Start address of the operating device's outputs<br>(Example: AB10) |
| DBOFFSET | 0 | DB offset which is added to the data block address of the variables<br>(Example: 0) |
| VAROFFSET | 0 | Variable offset which is added to the byte address of the variables<br>(Example: 0) |

#### 2.8.1.18 FC85 for the Operating Mode 'Asynchronous with Synchronization Impulse' with S7-300

This function decodes the data transferred by the operating device, and executes read and write accesses accordingly. The data width is 10 bytes. Accesses are made to the data sets which were parameterized in IBS CMD.

DBOFFSET and VAROFFSET also permit access to data blocks and byte addresses greater than 255.

**Parameter:**

```
Call FC 85PIN   :=  EB10POUT:=AB10DBOFFSET:=0VAROFFSET:=0
```

Table 2-26 Parameters for the FC85 function

| Parameter | Value | Description |
|---|---|---|
| PIN | EB10 | Start address of the operating device's inputs (Example: EB10) |
| POUT | AB10 | Start address of the operating device's outputs (Example: AB10) |
| DBOFFSET | 0 | DB offset which is added to the data block address of the variables (Example: 0) |
| VAROFFSET | 0 | Variable offset which is added to the byte address of the variables (Example: 0) |

#### 2.8.1.19 Example for the Operating Mode 'Asynchronous'

OB1 and OB100 for S7-300

When calling up the function FC20 (INIT_IB) in OB100, you must set the MODE parameter to zero (0).

**OB1**

```
...
Call FC 66
PIN             10
POUT            10
IBDB          DB20
DEST   DB10.DBX0.0
SOURCE DB10.DBX8.0
DBOFFSET         0
VAROFFSET        0
...
BE
```

**OB100**

```
UN M          10.0
S  M          10.0
R  M          10.1
CALL FC 20
IBDB            :=  DB20
COM_ADR         :=  256
DIAG_STATE      :=  0
DIAG_PARA       :=  0
FKN_START       :=  0
FKN_PARA        :=  0
FKN_STATE       :=  0
MEM_READ        :=  21
MEM_WRITE       :=  22
LOAD            :=  0
```

```
BOOT            :=  0
MODE            :=  0
TIMER_NR        :=  T1
SOURCE          :=  0
CONFIGURATION   :=  DW#16#8000000F
RET             :=  M10.1
BUSY            :=  M10.0
UN M         100.7
S  M         100.7
UN M          10.0
UN M          10.1
BE
```

OB1 for S7-400

**OB1**

```
...
Call FC 60
PIN        P#P10.0
DESTP#DB101.DBX0.0 Byte 8
SOURCEP#DB101.DBX8.0 Byte 8
POUT       P#P10.0
DBOFFSET         0
VAROFFSET        0
...
```

#### 2.8.1.20 Example for the Operating Mode 'Asynchronous with Synchronization Impulse'

OB1, OB40, OB100, FC50 and FC51 for S7-300

**OB1**

```
...
CALL FC 28
IBDB:=         DB20
READ_FC:=      FC50
WRITE_FC:=     FC51
CYCLE_OB:=     TRUE
RET:=         M14.0
...
BE
```

**OB40**

```
CALL FC 28
IBDB:=         DB20
READ_FC:=      FC50
WRITE_FC:=     FC51
CYCLE_OB:=    FALSE
RET:=         M12.0
BE
```

**OB100**

```
UN M          10.0
S  M          10.0
R  M          10.1
CALL FC 20
IBDB:=         DB20
COM_ADR:=       256
DIAG_STATE:=      0
DIAG_PARA:=       0
FKN_START:=       0
FKN_PARA:=        0
FKN_STATE:=       0
MEM_READ:=       21
MEM_WRITE:=      22
LOAD:=            0
```

```
BOOT:=           0
MODE:=           2
TIMER_NR:=      T1
SOURCE:=         0
CONFIGURATION:=DW#16#8000000F
RET:=        M10.1
BUSY:=       M10.0
UN M          10.0
UN M          10.1
BE
```

### FC50

```
UN M          10.0
S  M          10.0
CALL FC 21
IBDB:=        DB20
MODE:=           1
SOURCE:=         0
DEST_AREA:=      0
DEST_AREA_NR:=   0
DEST_OFFSET:=    0
DEST_LENGTH:=    0
RET:=        M10.1
BUSY:=       M10.0
CALL FC 65
PIN:=         EB10
POUT:=        AB10
DBOFFSET:=       0
VAROFFSET:=      0
BE
```

### FC51

```
UN M          11.0
S  M          11.0
CALL FC 22
IBDB            :=  DB20
MODE            :=  1
SOURCE_AREA     :=  0
SOURCE_AREA_NR  :=  0
SOURCE_OFFSET   :=  0
SOURCE_LENGTH   :=  0
DESTINATION     :=  0
RET             :=  M11.1
BUSY            :=  M11.0
BE
```

# A Index


C

Considerations for optimized data transmission 2-4

E

Error messages
INTERBUS MMICOM, Interface 06, 14, 18, 21, 76 ..... 2-14

I

Important notes ..... 1-1
Intended use ..... 1-1
INTERBUS MMICOM, Interface 06, 14, 18, 21, 76
2-1

M

MMICOM
MMI-COM profile ..... 2-1

O

Optimized data transmission ..... 2-8
Considerations ..... 2-4

P

Protocol parameters
INTERBUS MMICOM, Interface 06, 14, 18, 21, 76 ..... 2-7

S

Safety notes ..... 1-1
Symbols ..... 1-1
Syntax diagram
INTERBUS MMICOM ..... 2-9

T

Target group ..... 1-1

SÜTRON
visualization | control | communication

**Sütron electronic GmbH**
Kurze Straße 29
D-70794 Filderstadt
Phone: 0049 711 / 77098-0
Fax: 0049 711 / 77098-60
E-Mail: doku@suetron.de
Internet: www.suetron.com